Reducing the Human Cost of Grid Computing With glideinWMS

Igor Sfiligoi, Frank Würthwein,
Jeffrey Michael Dost, Ian MacNeill
University of California San Diego
La Jolla, CA 92093, USA
email: isfiligoi@ucsd.edu, fkw@ucsd.edu,
jdost@ucsd.edu, imacneill@ucsd.edu

Burt Holzman, Parag Mhashilkar
Fermi National Accelerator Laboratory
Batavia, IL 60510, USA
email: burt@fnal.gov, parag@fnal.gov

***Abstract*—The switch from dedicated, tightly controlled compute clusters to a widely distributed, shared Grid infrastructure has introduced significant operational overheads. If not properly managed, this human cost could grow to a point where it would undermine the benefits of increased resource availability of Grid computing. The glideinWMS system addresses the human cost problem by drastically reducing the number of people directly exposed to the Grid infrastructure. This paper provides an analysis of what steps have been taken to reduce the human cost problem, alongside the experience of glideinWMS use within the Open Science Grid.**

***Keywords**-Grid; glideinWMS; human cost*

I. Introduction

Over the past decade, the science community has been moving from dedicated, tightly controlled compute clusters to a widely distributed, shared Grid infrastructure in an effort to both increase the average equipment utilization and gather additional compute resources in times of need. One such Grid infrastructure is the US-based Open Science Grid (OSG) [1,2], an umbrella organization gluing together groups of scientists from many scientific domains. These groups are normally referred to as Virtual Organizations (VOs), since they have an internal structure. Each VO brings to the community both people and compute resources, with the understanding that their compute resources can be used by other VOs when not needed by the owning VO, and conversely that their users can access resources they don't own, when available.

This system has greatly benefited several VOs, but the early adopters have noticed that using the Grid can have a very high human cost. While the Grid is quite easy to use as long as everything works fine, when something goes wrong, it can take a significant amount of human time to debug and fix the problem. Given that the OSG currently encompasses O(100k) CPU cores distributed over O(100) geographic locations, having at least a few misbehaving nodes at any given time is pretty much a given. And with a community of O(10k) users, each broken node is likely to affect hundreds of users before being fixed. If each user were to spend even half an hour debugging the problem, the total human cost can easily exceed a week worth of time for each such event.

The glideinWMS system [3,4] attempts to reduce the human cost in two ways. It creates a dynamic overlay on top of Grid resources, thus insulating the final users from Grid problems, and it cleanly separates the VO policy handling from the actual Grid interfaces, allowing for a generic Grid-facing service, called a glidein factory, that further limits the exposure to the complexities of the Grid. To the best of our knowledge, this is the only system that supports that.

The glideinWMS has been in use on OSG with a shared glidein factory for over 2 years, and has proven to be a major success, drastically reducing the human cost of several VOs.

Section II provides an overview of the pilot paradigm, and the cost savings associated with it. Section III describes the cost savings due to the glideinWMS approach of separating VO policy from Grid submission. Finally, Section IV provides the analysis of the cost savings that OSG achieved in using the glideinWMS with a shared glidein factory.

II. Cost Advantage of Pilot Infrastructures

A pilot system [3] creates a dynamic overlay pool of compute resources on top of the Grid, as shown in Fig. 1. From the end user point of view, this overlay pool looks and feels exactly like a dedicated, tightly controlled compute cluster of the past, it is just a dynamic one, growing and shrinking depending on workloads and Grid resource availability.

Pilot infrastructures use two mechanisms to shield the users from Grid errors. The first and most important protection is provided by the pilots themselves; if a malfunctioning node kills the pilot before it is able to join the overlay pool, the users will never be aware of the existence of such node, preventing any error condition at its root. Starting the pilot is however not a sufficient condition to assure job success, since user jobs may need access to resources not needed by the pilot itself, e.g., scientific libraries, or they may need them in larger quantities, e.g., disk space. To account for that, most pilot system implementations, and in particular glideinWMS, allow for additional validation procedure to be run before joining the overlay pool; if even one test fails, the pilot aborts and never joins the pool. This allows for the overlay pool to be well behaved, at least within the limits of the tested properties.

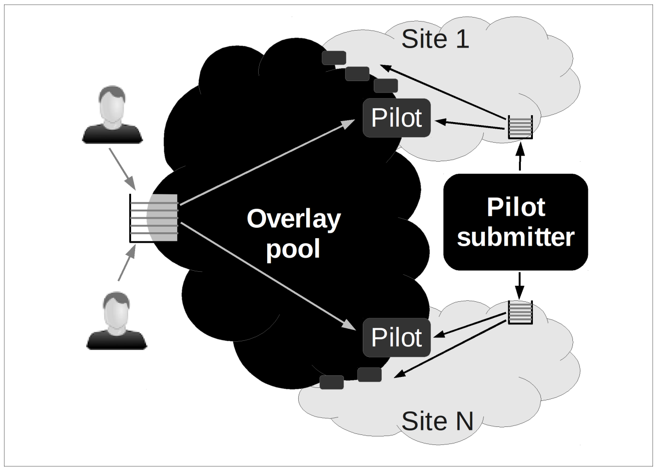

Figure 1. Schematic view of a pilot system

For the final user, the human cost of using this pool is thus comparable to using a truly dedicated compute pool. However, someone still has to create this overlay pool by submitting pilot jobs to the Grid. This pilot administrator will thus be exposed to the Grid-related errors affecting the pilot jobs themselves, and will be responsible for debugging them. While the human cost of this individual will obviously be much higher compared to the human cost of any individual user in the direct Grid submission paradigm, its cost is arguably still much smaller than the aggregate human cost of all the individuals.

There are two reasons for the cost savings. The first one is due to the difference in the type of jobs failing. Each user job is precious, so users have to spend some time recovering each and every one of them. Pilots are instead disposable, since they by themselves don't carry any useful payload, and any failure before an actual user job is started does not represent any loss of data, just reduced efficiency on the failing node. The human cost thus scales only with the number of failing nodes, not failing jobs. As shown in Table I, for a sizable OSG VO of O(1k) users running O(10M) compute jobs per month on O(1k) nodes, if even 1% of those jobs were to fail due to Grid problems, the use of a pilot infrastructure would reduce the effort from debugging O(100k) user jobs to debugging O(10) Grid nodes, thus decreasing the human cost by several orders of magnitude.

TABLE I. Comparison of debugging costs for a sizable OSG VO

	Direct submission	**Pilot system**
Metric (/month)	O(10M) jobs	O(1k) nodes
Error rate	O(1%)	O(1%)
Entities to debug	O(100k)	O(10)

The second reason is due to the difference in expertise. End users are typically not interested in computing, being scientists and viewing computing just as a tool, so they will likely spend a large amount of time trying to understand the occasional set of Grid-related problems. Pilot administrators can instead be IT professionals, who are well versed in debugging and fixing these kind of problems. Moreover, they will see similar errors with a much higher frequency, making the time-to-resolution dramatically shorter.

III. Importance of Partial Sharing in Pilot Infrastructures

The typical way of using pilot infrastructures is for each Virtual Organization to install a completely independent instance. This has been the approach of the early adopters of pilot infrastructures, such as the LHCb [5], CDF [6] and ATLAS [7] VOs.

The net result of this approach, however, is the proliferation of pilot administrators. Given that many Grid sites provide resources to many VOs, it also likely results in duplicate effort of debugging errors for pilots that happen to land on the same malfunctioning compute nodes. Offloading the operational load of many VOs to a single operations group would thus result in significant human cost savings, for the same reasons described in the previous section.

One of the reasons why early adopters did not go for a shared solution is that while sharing of a pilot instance is in theory possible, e.g., by simply allowing users from different communities to submit to the same overlay pool, in practice VOs cherish their autonomy, and will not delegate all control to a third party. As long as pilot submission is tightly integrated with the overlay pool operations, as it was the case for the solutions referenced above, partial sharing is not an option.

The glideinWMS addresses the above problem by clearly splitting the pilot infrastructure in two logical pieces, and thus separating the pilot submission from the operation of the overlay pool itself. The pilot submission is handled by one or more **glidein factories**, while the overlay pool is handled by the Condor batch system [8,9], with an additional process, called the **VO frontend**, providing the logic for requesting pilot submission from a glidein factory. Each glidein factory, in turn, can serve multiple VO frontends. The complete architecture is summarized in Fig. 2; please note that Condor pilots are labeled as **glideins**.

Using the glideinWMS, each VO operates its own Condor batch system instance and the associated VO frontend. Since almost all the policies are implemented in this layer, the VO maintains the full control of the overlay pool, thus retaining the look-and-feel of a dedicated, tightly controlled compute cluster.

A VO could also run a glidein factory, but it can instead delegate this activity to a third party without relinquishing any control of the system. The glidein factory is effectively a slave to the VO frontends, submitting pilots on their request. The added value of a glidein factory is mostly in the insulation of a VO frontend, and through it the associated Condor batch system, from the Grid world, providing Grid site specific configuration and validation, and handling all the Grid-related monitoring and error debugging. All of these activities are completely generic, and can be shared among any number of VOs.

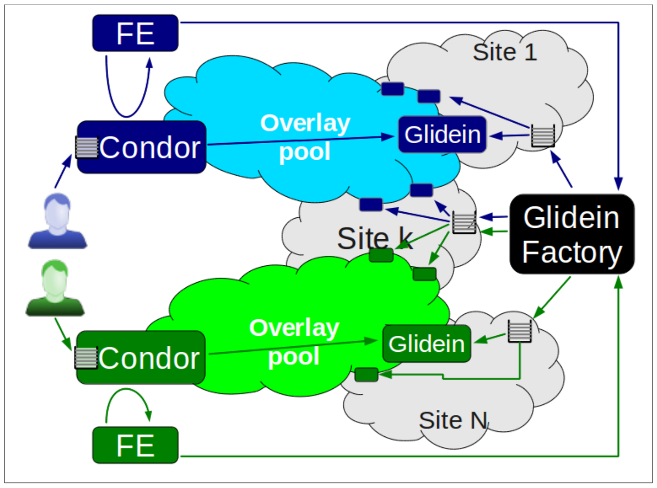

Figure 2. A glideinWMS glidein factory serving two VO frontends

One obvious concern in concentrating all operations to a single entity is that it may become the single point of failure. However, the glideinWMS architecture addresses this concern by allowing each VO frontend to be interfaced with multiple glidein factories, if so desired. While having more than one glidein factory will likely raise the overall cost of the system, it allows to hedge the risk of badly run services, scalability limits as well as complete service shutdowns.

As stated above, the cost savings of using a common glidein factory stem from the fact that many Grid sites provide resources to many VOs; pilots from many VOs will thus land on any malfunctioning or misconfigured worker node. Since the human cost scales with the number of failing nodes being debugged by a pilot administrator, having multiple pilot administrators debug the same node is obviously more expensive compared to a single team doing this task. A quantitative comparison is available in the next section.

IV. GlideinWMS in OSG

The Open Science Grid has been financing the operation of a glidein factory located at University of California San Diego (UCSD) since 2009, with additional contribution coming from the CMS experiment [10]. This instance is operated by three people on part-time basis, with an average effort of little less than one FTE. This glidein factory is open to all OSG VOs, and is currently used by 12 of them, varying in size from small campus-Grid groups to large world-wide communities.

The UCSD glidein factory submits pilot jobs to about 100 Grid sites; out of these, about 30% are used by multiple VOs, as shown in Fig. 3. Grid sites are selected mostly based on which VOs they support. The glidein factory operators obtain this information from multiple sources, including Grid information systems, VO-specific information systems and community knowledge. As far as possible, all information is cross-checked and all new Grid sites validated before being advertised to the served VO frontends. This effort invested in the early validation is usually orders of magnitude smaller than the effort that would be needed to debug misconfigured or malfunctioning sites after the fact, saving precious human time.

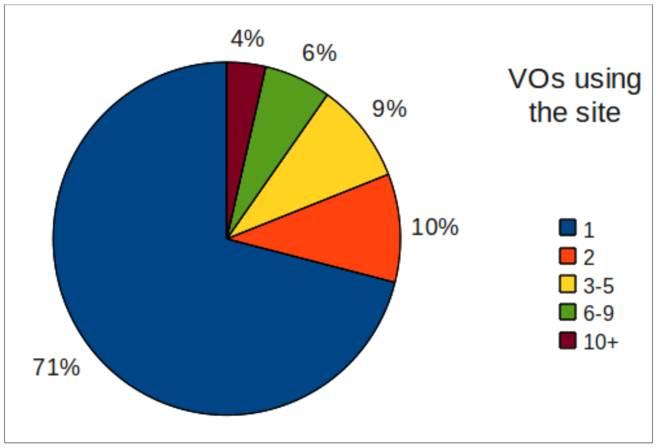

Figure 3. Fraction of OSG glidein factory Grid sites by number of VOs

As shown in Table II, in a typical week, this glidein factory submits about 200k pilot jobs, with about 130k or 65% running on shared Grid sites. Of all the submitted pilot jobs, about 25k or 12% fail the basic node validation, out of which about 22k running on shared Grid sites, yielding a slightly higher 16% error rate. About 25% of all human time is being spent on monitoring these kind of errors, identifying the root cause and collaborate with the affected Grid site administrators in resolving them. Given that significantly more than half of all failing pilots run on shared Grid sites, if each VO had to perform these functions by itself, it would have to spend at least 15% of a person's effort on this, which would result in at least 1.5FTE effort OSG-wide being dedicated to just Grid monitoring and debugging. Using a common glidein factory instance thus saves the OSG community well over a full time person time equivalent.

TABLE II. Weekly statistics of the OSG glidein factory

	All sites	Shared sites
Total glideins	200k	130k
Failing glideins	25k	22k

As can be seen, the major effort is currently not dedicated to day-to-day operations. Of the remaining time, about 40% is spent in helping the debugging of problems arising directly between Grid sites and the VO Condor batch system, another 20% writing tools to reduce the needed human effort in the long term, and the final 40% to help VOs to effectively use the glideinWMS. These numbers are also shown in Fig. 4.

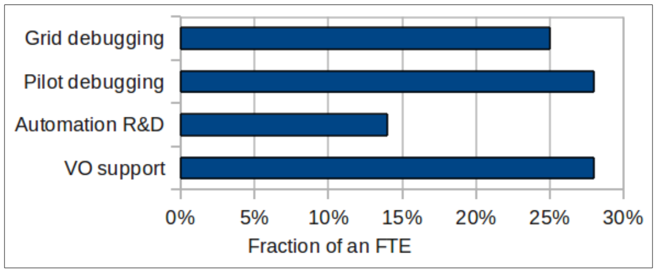

Figure 4. Allocation of effort at the OSG glidein factory

While problems arising from the use of Grid resources by the VO's Condor batch system is technically beyond the glidein factory control, the relevant error logs may not get propagated back to the VO, since the VO communication mechanisms are based on Condor itself. The glidein factory will instead always get them, since it is using the regular Grid mechanisms. The glidein factory operators are thus expected to monitor for these kind of errors as well.

The operators of the OSG-sponsored glidein factory instance also often take a leading role in solving such problems. These problems are often very similar in nature between different VOs; a typical example of such problems are firewall issues. As such, the glidein factory operators have extensive experience in debugging such errors, reducing the total human effort needed. This is especially important since these events, while relatively rare, often don't result in any obvious error messages in the logs, but require speculative thinking in order to be solved. Some of these speculative actions may be scriptable, so time is being invested into the R&D of such tools.

Finally, some of the OSG factory operators also help managing a CMS VO frontend and the related Condor batch system, so together with the experience of supporting several additional VOs from the glidein factory side, they are experts in troubleshooting every component of the glideinWMS system. As such, it is cost-effective to use these people to help all the OSG VOs in the configuration of their glideinWMS components. This does not mean they are involved in day-to-day operations, but they do advise on major configuration decisions.

The number of VOs supported by the OSG glidein factory has been gradually increasing with time. In this period, we noticed that new VOs typically require significant hand-holding, both in terms of configuration help as well as Condor problems on Grid resources during the initial setup period and during major changes in their operation mode, but require relatively little effort most of the remaining time. The human time required by the glidein factory operations team has thus been pretty much constant for all but the initial few months of the glidein factory lifetime, and is expected to significantly decrease once the influx of new VOs slows down.

TABLE III. FTE cost estimates for glideinWMS use in OSG

	Shared factory	VO provided factory	
		Per VO	OSG-wide (12 VOs)
Grid debugging	25%	15%	180%
Pilot Debugging	28%	15%	180%
Automation R&D	14%	10%	120%
Total	67%	40%	480%

The actual cost savings of using a shared OSG glidein factory are difficult to measure, since most VOs using it switched directly from direct submission to the shared-factory pilot paradigm. We thus made an educated guess about the operational costs a typical OSG VO would incur by running its own glidein factory, and presented them in Table III. Given that more than half of all pilots run on shared Grid sites, we estimated that the per-VO cost of both Grid and pilot debugging would scale approximately at the same rate; the automation R&D would instead likely be almost the same as in the shared glidein factory scenario, although the shared glidein factory does need to produce more complex tools. As can be seen, we estimate that the OSG VOs would each use about 40% of an FTE, for an OSG-wide total of about 5 FTEs. This is significantly higher than the 2/3 FTE currently being used by the shared glidein factory.

V. Conclusion and Future Work

Using Grid resources directly can have a high human cost. While the Grid is quite easy to use as long as everything works well, when something does go wrong, it can take a significant amount of human time to debug and fix the problem. Several OSG Virtual Organizations have thus switched to the use of glideinWMS, which allows for significant cost savings.

The major cost savings come from glideinWMS being a pilot system, i.e. creating a dynamic overlay pool of compute resources on top of the Grid. This shields the end users from Grid errors, and delegates their debugging to a dedicated team of professionals. Furthermore, to achieve savings across different VOs, the glideinWMS architecture separates the pilot submission services from the VO logic, shielding even the VO administrators themselves from the Grid, and allowing for the outsourcing of the Grid error handling to an experienced operations team.

The Open Science Grid has thus invested into a common glidein factory instance, creating an expert operations team that handles the Grid-related monitoring and debugging tasks for all the interested VOs. This allows these VOs to drastically reduce the human effort needed, resulting in global savings of several full time persons time compared to running the complete pilot infrastructure themselves. The cost savings compared to direct Grid submission can instead be counted in tens of FTE, given the thousands of scientists using the Grid resources.

Moreover, the outsourcing of Grid-related activities also contributes to a much better user experience, since most Grid-related problems are caught before the users are exposed to them, and the remaining ones get solved quickly thanks to the experience of the dedicated glidein factory operations team. This contributes to a greater usage of Grid resources by scientists who would otherwise avoid them, due to the high human cost involved.

The system has served OSG well, both in terms of effectiveness and human cost, and is expected to continue to operate in the foreseeable future, with most OSG VOs eventually using it. The only major operational change currently planned is the creation of a second glidein factory instance at a different location, for high availability reasons. While this is expected to slightly increase the operations costs, it is a highly desirable step now that a large community depends on it.

ACKNOWLEDGMENT

This work is partially sponsored by the US Department of Energy under Grant No. DE-FC02-06ER41436 subcontract No. 647F290 (OSG), and the US National Science Foundation under Grants No. PHY-0612805 (CMS Maintenance & Operations), and OCI-0943725 (STCI).

REFERENCES

[1] R. Pordes et al., “The open science grid,“ J. Phys.: Conf. Ser., vol. 78, 012057, pp. 1-5, 2007, doi: 10.1088/1742-6596/78/1/012057.

[2] “Open Science Grid home page,” http://www.opensciencegrid.org/, Accessed June 2011.

[3] I. Sfiligoi et al., "The pilot way to grid resources using glideinWMS," CSIE, WRI World Cong. on, vol. 2, pp. 428-432, 2009, doi: 10.1109/CSIE.2009.950.

[4] “glideinWMS,” http://tinyurl.com/glideinWMS, Accessed June 2011.

[5] A. C. Smith and A. Tsaregorodtsev, “DIRAC: reliable data management for LHCb,” J. Phys.: Conf. Ser., vol. 119, 062045, pp. 1-6, 2008, doi: 10.1088/1742-6596/119/6/062045.

[6] S. Belforte et al. “GlideCAF: A late binding approach to the grid,” Proc. Comp. in High Ener. and Nucl. Phys. (CHEP2006), 2006, id 147, http://indico.cern.ch/materialDisplay.py?contribId=147&sessionId=8&materialId=paper&confId=048, Accessed June 2011.

[7] T Maeno, “PanDA: distributed production and distributed analysis system for ATLAS,” J. Phys.: Conf. Ser., vol. 119, 062036, pp. 1-4, 2008, doi: 10.1088/1742-6596/119/6/062036.

[8] D. Thain, T. Tannenbaum, and M. Livny, “Distributed computing in practice: the Condor experience,” Conc. and Comp.: Practice and Experience, vol. 17, issue 2-4, pp. 323–356, 2005, doi: 10.1002/cpe.938.

[9] “Condor project homepage,” http://www.cs.wisc.edu/condor/, Accessed June 2011.

[10] The CMS Collaboration et al. “The CMS experiment at the CERN LHC,” J. Inst, vol. 3, S08004, pp. 1-334, 2008, doi: 10.1088/1748-0221/3/08/S08004.